Scan Statistics for the Online Detection of Locally Anomalous Subgraphs

Joshua Neil[†], Curtis B. Storlie [††], Curtis Hash[†], Alexander Brugh[†], Mike Fisk[†]
[†] Advanced Computing Solutions [††] Statistical Sciences Group (CCS-6)
Los Alamos National Laboratory

March 26, 2012

Abstract

Identifying anomalies in computer networks is a challenging and complex problem. Often, anomalies occur in extremely local areas of the network. Locality is complex in this setting, since there is an underlying graph structure. To identify local anomalies, a scan statistic is introduced for data extracted from the edges of a graph over time. Two shapes are described for capturing locality in the graph: the star and the k-path. The use of the path as a scan window is novel. Both of these shapes are motivated by hacker behaviors observed in real network attacks. These shapes are enumerated over the entire graph, over a set of sliding time windows. Local statistics in each window are compared with their historic behavior to capture anomalies. These local statistics are model-based, and two models motivated by network flow data are presented. Data speeds on larger networks require online detection to be nimble. An anomaly detection system was therefore designed that achieves far better than real-time analysis speed, as applied to Los Alamos National Laboratory's (LANL's) entire internal network (∼20,000 hosts). In addition, a result from our analysis of an actual month's worth of data from LANL's internal network is given.

*Keywords:*Dynamic Graph, Anomaly Detection, Network Monitoring, Path

1 Introduction

In this work, we consider the problem of detecting locally anomalous activity in a set of time-dependent data having an underlying graph structure. Our motivational data set, a large computer network, yields graphs on the order of 20,000 nodes and 90,000 edges, every 30 minutes. A method is described to identify attacks occurring in this network in real-time. In particular, we are interested in identifying local regions within the graph that have deviated from historic behavior in some window of time.

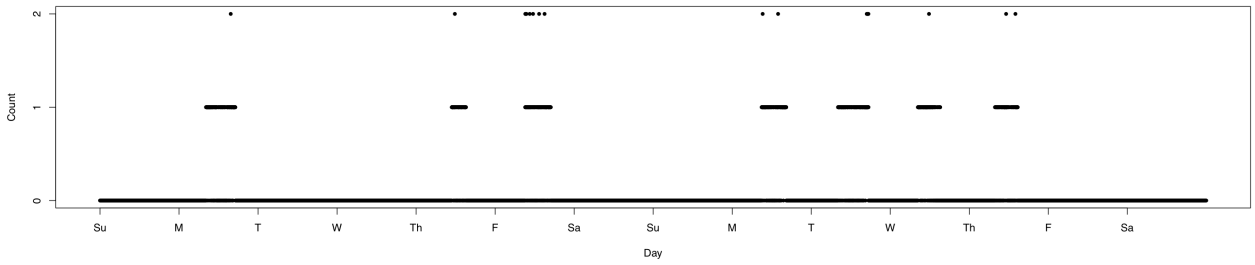


Figure 1: Plot of the communications between two machines in Los Alamos National Laboratory's (LANL's) internal network. The plotted value is number of NetFlow connections per minute. The flows originate at a specific user's machine, and the destination is a server providing virtual desktop services.

The data used to illustrate this method was obtained from NetFlow records [Phaal et al., 2001, Lyons, 1997, Brownlee et al., 1999] gathered from one of LANL's internal networks over 30 days, starting January 30, 2010. In terms of a graph, Internet Protocol (IP) addresses define nodes and any communications between IPs define the existence of a directed edge between those IPs in the graph (resulting in a total of 558,785 edges). NetFlow data consists of connection layer information, from which the number of connections originating at the source of the directed edge and bound for the destination is extracted and binned by minute to produce the data set.

A plot of one of these edges is given in Figure 1. It should be understood, however, that there is an enormous variety of behavior amongst edges in computer networks, and this plot obviously does not capture that variety. For brevity, however, we omit a more comprehensive description of the edge data. On the other hand, this edge does represent many edges where a human actor is present on the originating machine.

To help clarify the approach, an anomalous scenario that is of interest is described. Consider an attack by a hacker on a computer network. A common initial stage of the attack is to infect a machine on the network using malicious software. One method for initial infection is known as a phishing attack, where an email that includes a link to a malicious website is sent to a set of users on a network. When the user clicks on the link, their machine becomes infected, giving the

attacker some form of access to the machine.

The attacker generally cannot dictate which machine is infected, and the initial host is usually not the ultimate target of the attack, if there is a predefined target. Instead, the hacker may wish to move to other machines in order to locate and exfiltrate valuable data, escalate privileges, or to establish broad presence in the network for later exploitation. Therefore, from this initial host, the attacker may proceed to other hosts, hopping from one to the next; see Figure 2.

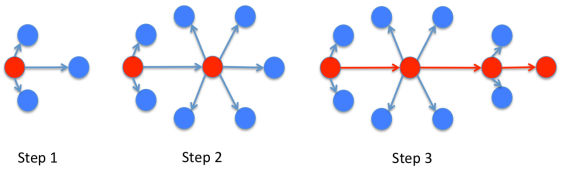


Figure 2: A traversal attack. Step 1: Initial infection and local search. Step 2: First traversal has occurred, and further search is performed. Step 3: A full traversal has occurred. This shape is denoted as a *caterpillar*.

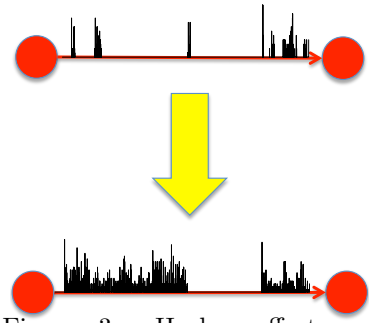


Figure 3: Hacker effects on the time series of data between two machines. Upper: normal activity between two machines. Lower: hacker acts *in addition* to normal activity to create anomalous edge.

As the attacker traverses the network, he creates anomalous activity in the time series of communications along each edge he traverses (see Figure 3). The union of these anomalous edges in some interval of time describes the anomaly that we seek to detect.

1.1 General Approach and Organization of This Work

We are interested in testing the null hypothesis that all edges in the graph are behaving as they have historically, versus the alternative that there are local shapes of altered activity among the edges. To accomplish this goal, a method has been developed, based on scan statistics, to examine each of these shapes in the graph over sliding windows of time.

Scan statistics have been widely used to detect local clusters of events [Glaz et al., 2001, Kulldorff, 1997, Naus, 1982, Loader, 1991]. The idea is to slide a window over a period of time and/or space, calculating a local deviation statistic. The most extreme of these is known as the scan statistic, which is used to decide if there is any deviation from historic behavior in the local window.

In this work, the focus is on two basic shapes in the graph: stars and paths. While the use of stars as a scan object has been examined [Priebe et al., 2005], the path scan object is novel to the literature. The case is made for basing the scan statistic on paths to discover anomalous traversal using both simulated examples and the internal network at LANL. It is important to note, that it is the *union* of these basic shapes which ultimately forms the detection region (see Section 2.2). The basic shape is only the building block upon which the detected anomalous shape is based.

The unions of both path and star anomalies (see Section 2.3) have been observed in attacks in LANL's computer network. A star anomaly is indicative of a hacker using a compromised machine

to attempt to connect only to machines it has immediate access to, creating anomalies on multiple edges emanating from the compromised host. A path anomaly indicates a more subtle attack, which is a sequence of traversals from each host in the path to the next. The caterpillar anomaly, which is a mixture of stars and paths, is also discussed.

This approach was designed to monitor a computer network in real time, and any scheme applied to computer network data at an enterprise-level (20,000 or more individual IP addresses) needs to be fast. Yet, in order to identify highly local anomalies, the system needs to monitor many small windows simultaneously. A prototype monitoring system has been designed and implemented that is capable of examining an extremely large number of local objects in a corporate-sized network in real-time.

1.2 Related Work

Fast statistical anomaly detection on streaming data has become an important area of research given the proliferation of data over the past few decades, and the need to detect quickly the event that a process has changed significantly from past behavior. Applications can be found in many areas including engineering [Chandola et al., 2009], computer science [Forrest et al., 1996], and, specifically, in communications networks [Mukherjee et al., 1994, Lambert and Liu, 2006, Yeung and Ding, 2003, Chandola et al., 2009].

In many cases, the data can be represented as a graph [Kolaczyk, 2009]. Nodes represent actors sending and receiving data, and edges represent communications between nodes. Anomalies can be detected in the changes to the structure of the graph [Priebe et al., 2005, Collins and Reiter, 2007, Noble and Cook, 2003]. These methods tend to lack fine-grained locality, which is addressed by using path scans in Section 2.3. Another approach would be to aggregate all communications emanating from each node, and consider nodes to be independent [Yeung and Ding, 2003, Mukherjee et al., 1994]. However, in many networks, node behavior may not be independent, and different edges may have significantly different behavior over time. In addition, traversal cannot be captured by only analyzing node behavior. Thus, in the problem considered here, modeling each edge is desirable.

In Heard et al. [2010], individual edges are modeled, and a Bayesian testing framework is proposed to test the anomalousness of each edge, without consideration of other local edge anomalousness. These edges are then passed to a secondary analysis that looks for centrality in the graph constructed from the edges that were detected in the initial pass. Centrality in the anomalous edge graph can be detected in this way, but simultaneously testing multiple local sets of edges will have increased power to detect locally anomalous behavior. For example, if two anomalous edges were

connected by a non-anomalous edge, this possible traversal path would likely be missed by the technique in Heard et al. [2010], but is a valid anomaly in many settings. In addition, when data speeds are high, a fully Bayesian treatment may pose computational difficulties, unless the model is parsimonious enough for sequential Monte Carlo [Doucet et al., 2001].

Scan statistics have been used to detect anomalies in the Enron email graphs [Priebe et al., 2005]. While they scan for anomalies in the out degree of every node in the graph over time (a star shape), this method examines any general shape the user wishes to define. One conclusion of this work is that paths capture very general anomalous shapes, but star anomalies are too global to identify very localized anomalies.

Paths have been examined in the context of vehicular traffic in Lu et al. [2009], using a similarity metric to compare paths, and then clustering to find outliers. This method, however, assumes that path values that can be clustered are observed. On the contrary, in this work a statistically rigorous method to infer anomalous shapes from the network without any prior knowledge about traversal by individual actors is proposed.

Many of the methods mentioned above are intended for much smaller graphs than this method proposes to address. We have a data set that is difficult to come by: a record of all of the communications between individual computers on a large corporate-sized network. This data is recorded continuously with second level resolution, or better, and has been archived for the past decade in some cases. The objects (paths) monitored number in the hundreds of millions per 30 minute window, and the proposed method is able to analyze these objects in under 5 seconds. The sheer size of this endeavor, we believe, separates it from most other work on graph-based scan statistics, and anomaly detection in general. The telephone network literature [Lambert and Liu, 2006, Lambert et al., 2001] alone monitors larger networks than ours, but not in a graph based setting, instead monitoring aggregation points in the network.

2 Methodology

In this section, the methodology behind scanning for local anomalies in a graph over time is described. Windowing in this space is discussed, followed by the definition of the scan statistic. Finally, the star and the k-path are discussed.

2.1 Windows in the Cross Product Space

Sets of windows in the $Time \times Graph$ product space are described, which are defined as follows. Let $G = (V, E)$ be a graph with node set V and edge set E. For each edge $e \in E$,

at discrete time points $t \in \{1, \ldots, T\}$, a data process $X_e(t)$ is observed. Denote the set of all possible time windows on all edges e over discretized time intervals $(s, s+1, \ldots, k)$ as $\Omega = \{[e, (s, s+1, \ldots, k)] : e \in E, 0 \leq s < k \leq T\}$.

The set of all subsets of edge-time windows, $\Gamma = \{\{w_1, w_2, \ldots\} : w_j \in \Omega\}$, is usually very large. However, we are normally interested in only a subset, $\Gamma_s \subset \Gamma$, that contains locality constraints in time and in graph space, and we therefore restrict our attention to sets of windows $\gamma \in \Gamma_s$. For example, Figure 4 shows a window γ coming from a Γ_s given by the set of all 3-paths (more formally defined in Section 2.3), where the time component is a set of equal intervals on each edge in the path.

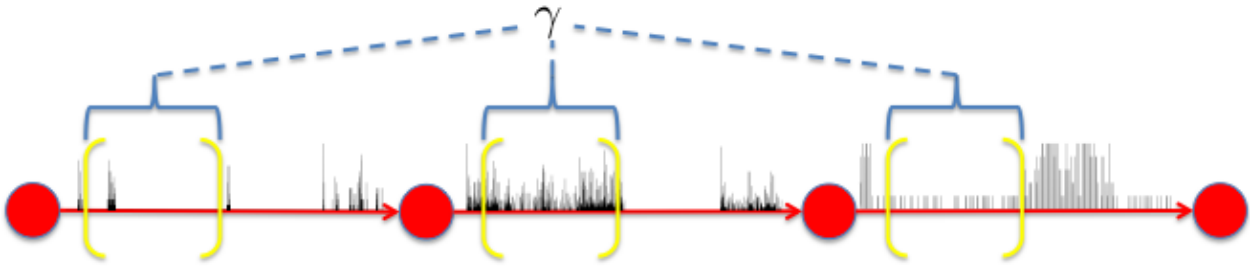


Figure 4: The windows γ are sets of edges along with time intervals on each edge. In this example, a 3-path with a set of equal time intervals (indicated by brackets) is given.

For convenience, denote $\boldsymbol{X}(\gamma)$ as the data in the window given by γ. Assume that for any time point t and edge e, $X_e(t)$ can be described with a stochastic process (specific examples are given in Section 3) with parameter function given by $\boldsymbol{\theta}_e(t)$. Denote the values of the parameter functions evaluated in the corresponding set of windows γ by $\boldsymbol{\theta}(\gamma)$. Finally, denote the likelihood of the stochastic process on γ as $\mathcal{L}(\boldsymbol{\theta}(\gamma)|\boldsymbol{X}(\gamma))$.

2.2 A Scan Statistic for Windows in the *Time* × *Graph* Space

The hypothesis test revolves around whether the data in a window could have been produced by some known function of the parameters $\hat{\boldsymbol{\theta}}(\gamma)$, versus alternatives indicating that the parameters have changed. That is, given that $\boldsymbol{X}(\gamma) = \boldsymbol{x}(\gamma)$ is observed, we would like to test whether $H_0 : \boldsymbol{\theta}(\gamma) = \hat{\boldsymbol{\theta}}(\gamma)$ against alternatives that can be formed by restricting the overall parameter space, $\boldsymbol{\Theta}$, to a subset $\boldsymbol{\Theta}_A \subset \boldsymbol{\Theta}$. The generalized likelihood ratio test statistic (GLRT) is a natural statistic to use. Let

$$\lambda_\gamma = -2 \log \left(\frac{\mathcal{L}\left(\hat{\boldsymbol{\theta}}(\gamma)|\boldsymbol{x}(\gamma)\right)}{\sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}_A} \mathcal{L}\left(\boldsymbol{\theta}(\gamma)|\boldsymbol{x}(\gamma)\right)} \right) \tag{1}$$

The size of λ_γ depends on the number of parameters being tested in the window, making it difficult to use directly. To address this issue, λ_γ is normalized by converting it into a p-value, p_γ. These

p-values are discussed in detail in Section 3.6.

To scan for anomalies in the $(Time \times Graph)$ product space, all windows γ are examined, keeping track of the scan statistic $\Psi = \min_\gamma p_\gamma$. In practice, thresholding must be done on the set of p-values, and so more windows than just the γ corresponding to Ψ should be considered. For online monitoring, a threshold on the p-values is set to control the false discovery rate [Benjamini and Hochberg, 1995]. This threshold setting is described in more detail in Section 3.7, but we emphasize that generally, when a detection occurs, a set of windows (not just one) are detected, and so the union of these windows is the detected anomaly produced by the system.

So far, the description of Γ_s, the set of sets of windows to be scanned, has been very general. In the next section, more specific window shapes for detecting anomalous computer network behavior are described.

2.3 Local Shapes: Stars and Directed k-Paths

The approach described in Section 2.2 can be used for batch (retrospective) or online (prospective) processing. But graphs are combinatorial in nature. For a fully connected graph with n nodes, the number of subgraphs is $2^{n(n-1)}$. This motivates the need for an extremely restricted set of graph windows, especially in the online setting. To this end, windows are constructed that are appropriate for identifying specific shapes of anomalies.

Directed k-Paths. Since a primary motivating example is that of hacker traversal in a computer network, we suggest a specific type of subgraph for online monitoring: directed *k-paths*. A directed k-path is a subgraph of size k, which has diameter k. Here, size is the number of edges in a graph, and diameter is the greatest distance between any pair of nodes. Informally, this means that a k-path is a sequence of edges where the destination node of the current edge in the sequence is the source node of the next edge in the sequence, and so on.

The k-path has the advantage that it captures the core of many network attacks, since the attack is described by a path through the network, with additional edges as "fuzz" around this core path (as was depicted in Figure 2). This attack shape has been observed in actual attacks on LANL's network, and is therefore an important focus of this work. In addition, the k-path is highly local, allowing for the detection of very small anomalies. Finally, unions of k-paths form very general subgraphs, the only restriction being that the union graph's diameter must be at least k.

In the simulation and real data studies described in Sections 4 and 5 respectively, 3-paths were chosen. 3-paths have the advantage of locality, but are also large enough to capture significant

traversal. But in order to scan every 3-path in the network graph, they must first be enumerated. This can be non-trivial for many graphs. In a fully connected graph with n nodes, eliminating cycles and back edges, there are $n(n-1)(n-2)(n-3)$ 3-paths. In reality, our network graph is much less connected. However, in a 30 minute window of time, if only edges with non-zero activity in that window are included, a graph that contains around 17,000 nodes, 90,000 edges, and 300 million 3-paths is obtained. While the entire set of $n(n-1)(n-2)(n-3)$ possible 3-paths are effectively scanned, anomaly measures on paths with an edge that has no activity in the current time window are not calculated. Since a hacker needs to make at least one communication to traverse an edge, no activity on an edge indicates no traversal over that edge, and a path containing the edge should therefore not be considered anomalous (in the time window of interest).

Due to the large number of 3-paths, it is important to be able to enumerate paths quickly if we hope to maintain a near real-time response capability. A fast, parallel C library has been written for this purpose. At the core of the library is an algorithm that enumerates paths, a description of which can be found in the Supplementary Material. This algorithm uses very little memory, and is trivially parallelizable. In the simulations discussed in Section 4 and the real data example discussed in Section 5, enumeration and testing using the models described in Section 3 on 30 minute windows consisting of roughly 300 million paths was accomplished in under 5 seconds per window, using a 48 core commodity machine. This allows us room to add complexity to the models, and handle larger graphs than the already sizable graphs currently being analyzed, while keeping up with real-time data streams.

Stars. Stars are another interesting shape for monitoring communication networks. Stars are defined as the set of edges whose source is a given central node; see Figure 5. While these shapes are not very localized, especially for high out-degree nodes, they still pick up star-type anomalies rather well.

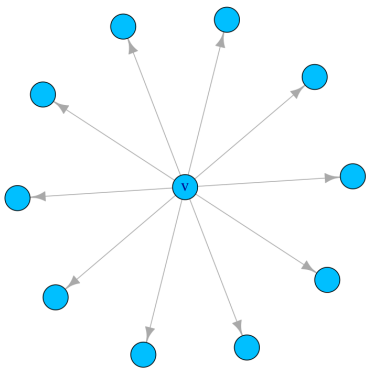


Figure 5: Example Out-Star, centered at node v.

It will be seen that paths have the ability to describe more subtle anomalies than star windows, but star windows generally outperform paths on large star anomalies.

Time Intervals. In this work, the time component consists of the same interval of time over every edge in the graph window. This will detect anomalies that occur in the same time window for each edge in the shape. There are more elaborate options such as telescoping time windows [Djidjev et al., 2011], which are useful if we wish to cater to specific protocols like the Secure Shell protocol.

3 Modeling, Estimation, and Hypothesis Testing

In Sections 4 and 5, the scanning procedure is illustrated using data extracted from LANL's internal network. The data is organized in the form of edges – the sequence of counts of communications between two individual machines. It is desirable to model at the resolution of edges, not at the resolution of the shapes γ, for computational reasons. Therefore, a study of the correlations among edges connected in a path is performed to justify an assumption of independence between edges. Following that, a description of two models motivated by the distribution of data on edges over time, including estimation, hypothesis testing, p-value calculation and thresholding is given.

3.1 Switching Processes in Edge Data

As can be seen in Figure 1, it is common in computer network data to observe a switching process. Intuitively, for many edges, this switching is caused by the human presence on the network. If a user is present at a machine, she may make non-zero counts on edges emanating from that machine. But in many minutes, even though the user may be present, she may not be making non-zero counts on this edge, since she may be communicating with some other machine, or not using the network at all. It is only clear that when she is not there, zeros will be observed on this edge.

This presence/absence induces a switching process between a purely zero count emission and a higher activity count emission. While, intuitively, there will be higher counts in the middle of the day than at night, in this work homogeneous models are used for the sake of simplicity. The simulation models were chosen to approximately reflect the data, but the focus here is on the method of scanning, and not on modeling of edges. More sophisticated models specifically capturing diurnal and weekly trends are a subject of further work.

3.2 Independence of Edges in a Path

In order to scan for anomalous shapes, it is necessary to have models that describe the behavior of the data in the window γ under normal conditions. The number of enumerated subgraphs tends to scale exponentially with the number of nodes, however, and an assumption of independence among the edges in the shape facilitates scaling the computations required to process graphs at line speeds, under reasonable memory requirements. This is because edge independence only requires models (and the storage of their parameters) for each edge, whereas non-independence might require models for each *shape*, of which there may be many hundreds of millions, if not billions. Therefore, an examination of the assumption of independence among edges connected in a path is given below.

This study utilized LANL NetFlow records over a 30 day period. For each edge, the data consists of a per-minute recording of the indicator variable which is 1 if there is activity on that edge in that minute and zero otherwise. This results in a sequence of 40,320 binary values for each edge. The sample correlation between pairs of edges connected in a 2-path was measured, to determine how much linear relationship is present within paths. In this data set, there were a total of 311,411 2-paths.

In Figure 6, a plot of the empirical cumulative distribution function for the R^2 statistics is given. Half of all squared correlations were less than 0.002%, and only 1 in 1000 had R^2 value of larger than 6%. Those 2-paths with evidence of correlation will need to be examined further, and in future work we intend to model dependencies between edges as needed. However, as indicated by Figure 6, correlation between connected edges in a path is rare. Therefore, for the rest of this study, edge independence is assumed, making computation and storage substantially less burdensome.

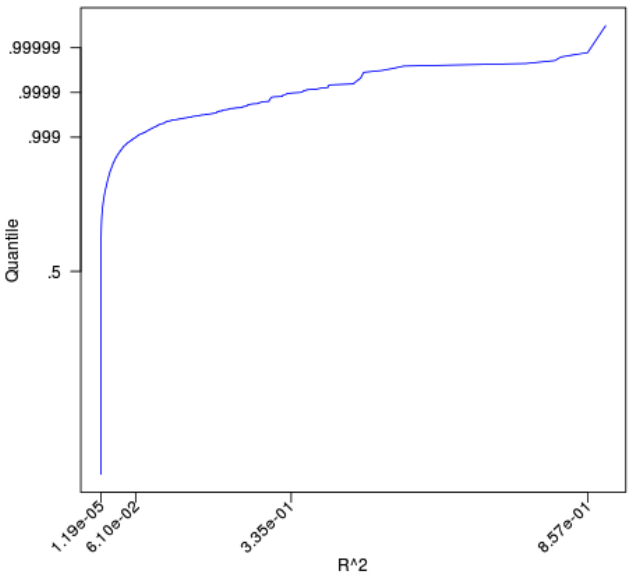


Figure 6: Empirical CDF of the squared correlations between pairs of edges connected in a 2-path.

Note that under the independence assumption, the path GLRT in (1) can be expressed as

$$\lambda_\gamma = \sum_{e \in \gamma} \lambda_e \tag{2}$$

where λ_e are the GLRT scores on each edge in window γ. Thus to compute λ_γ for all $\lambda \in \Gamma_s$, we only need to compute λ_e for each edge e, and combine these scores using simple addition.

3.3 Model Descriptions

The first and simplest model is a two-state *observed Markov model* (OMM), which emits a binary value denoted B_t. If there was a non-zero count in time bin t, then $B_t = 1$, otherwise $B_t = 0$. This model has two parameters, $p_{01} = P(B_t = 1 | B_{t-1} = 0)$ and $p_{10} = P(B_t = 0 | B_{t-1} = 1)$. Let $b_1, \ldots, b_N$ be the observed values of $B_1, \ldots, B_N$ for a given edge. Then the likelihood is given by

$$\mathcal{L}(p_{01}, p_{10} | b_1, \ldots, b_N) = (1 - p_{01})^{n_{00}} p_{01}^{n_{01}} p_{10}^{n_{10}} (1 - p_{10})^{n_{11}} \tag{3}$$

where n_{ij} is the number of times that the consecutive pair $(b_t = i, b_{t+1} = j), i, j \in \{0, 1\}$ was observed in the data. The initial state is assumed to be fixed and known. While this model captures the switching behavior seen in Figure 1, it ignores the distribution of non-zero counts, and also does not allow for zeros to be produced in the high state.

To address these shortcomings, a *two-state hidden Markov model* (HMM) [Rabiner, 1989] is employed. This model has a degenerate distribution at zero for the low state, and a negative binomial emission density in the high state. Negative binomial emission densities do not suffer from the equidispersion property of the Poisson [Pohlmeier and Ulrich, 1995], and a good justification for using them to monitor for anomalies in network counts is given in Lambert and Liu [2006]. Note that this model is similar to the hurdle models described in Heard et al. [2010] and elsewhere, with one important distinction: the high-state is allowed to emit zeros. We believe that this is important in modeling our data. Again, referring to Figure 1, zero counts are interspersed with the non-zero data, but are still clearly a part of the 'active' state. Intuitively, think of the active state as "the user is present at the machine," and therefore likely to make communications, not as "the user is making a communication on this edge."

Following the notation in Rabiner [1989], the observed counts, O_t, follow a "hidden" two-state Markov process, call it Q_t. The transition parameters are given by $p_{01} = P(Q_t = 1|Q_{t-1} = 0)$ and $p_{10} = P(Q_t = 0|Q_{t-1} = 1)$. The emission densities in each state are parameterized as $d_0(O_t) = P(O_t|Q_t = 0) = I(O_t = 0)$ and $d_1(O_t) = P(O_t|\mu, s, Q_t = 1) = NB(O_t|\mu, s)$ where $I(\cdot)$ is the indicator function and $NB(\cdot|\mu, s)$ is the Negative Binomial density function with mean μ and size s. The likelihood is given by

$$\mathcal{L}(p_{01}, p_{10}, \mu, p|O_1, \ldots, O_N) = \sum_{q_1} \cdots \sum_{q_N} d_{q_1}(O_1) p_{q_1 q_2} d_{q_2}(O_2) \cdots p_{q_{N-1} q_N} d_{q_N}(O_N) \tag{4}$$

More details are given in the Supplementary Material.

3.4 Pooling and Estimation

The exact number of edges with any non-zero activity seen in the month of training data was 558,785 edges, and parameter estimates were required for each of these. But many of these edges were very sparse, and, therefore, there was not much opportunity to observe high state counts. To perform the estimation, edges were pooled together according to μ_e, the average number of non-zero counts per day for each edge (averaged over all 30 days). Two Edge Types are defined:

- *Edge Type I* ($\mu_e \geq 1$) consists of those 252,165 edges (45%) for which there were at least 30 time bins with activity during the 30 days, and hence there was sufficient data to estimate an individual model. The maximum likelihood estimates (MLEs) are used to estimate the parameters on each of these edges.

- *Edge Type II* ($\mu_e < 1$) consists of the remaining 306,620 edges (55%). These edges will share a common parameter set, in order to borrow information across very sparse data. The set of edges $\tilde{e}$ such that $\mu_{\tilde{e}}$ was among the 1,000 largest μ_e values in Edge Type II is then extracted. Parameters on each of these edges are estimated, and the mean of these parameter vectors is the parameter vector used for the common edge model of Type II edges. Since the tests are focused on increased activity, taking the largest 1,000 μ_e will ensure that the models are not overly sensitive on these low count edges.

Maximum likelihood estimates for the parameters of the OMM are given by $\hat{p}_{01} = \frac{n_{01}}{n_{00}+n_{01}}$ and $\hat{p}_{10} = \frac{n_{10}}{n_{10}+n_{11}}$.

Maximum likelihood estimates of the HMM parameters have no closed form, so an expectation-maximization (EM) approach is used to maximize (4). Details are given in the Supplemental Material.

Estimation of 252,165 edges was somewhat intensive. Our C-code estimates edges at a rate of approximately 10.6 edges per minute per cpu, using a 48-core commodity machine, taking a total of just over 8 hours to estimate parameters for all edges. This is quite sufficient for our needs, since the full parameter estimation scheme does not need to be performed/updated at the time scale of the scan windows in practice. Rather, this updating can be done nightly or weekly, for example.

3.5 Alternative Hypotheses

In order to obtain a GLRT, the overall parameter space needs to be restricted to allow for alternatives which reflect the types of hacker behavior we wish to detect. These are intentionally kept general, in order to catch a variety of behaviors. We postulate that hacker behavior causes increases to the MLEs of parameters governing the models. This is due to the fact that the hacker must act *in addition to* the normal behavior on that edge. Specifically, referring to the OMM, we propose that hacker behavior causes an increase in the probability of transitioning from the inactive to the active state:

$$H_0 : p_{01} = \hat{p}_{01} \quad \text{versus} \quad H_P : p_{01} > \hat{p}_{01} \tag{5}$$

where $\hat{p}_{01}$ is the historic MLE.

In the HMM setting, there are more options. Three combinations of parameter changes will be tested:

$$H_P : p_{01} > \hat{p}_{01}, \quad H_M : \mu > \hat{\mu}, \quad H_B : p_{01} > \hat{p}_{01} \text{ and } \mu > \hat{\mu} \tag{6}$$

In each case, the null hypothesis is that the parameter or two-parameter pair is equal to its historic MLE value.

3.6 P-value Calculation

We seek a p-value for the observed GLRT statistic, λ_γ. Under mild regularity conditions, the GLRT is asymptotically χ^2 with degrees of freedom equal to the number of free parameters in $\mathbf{\Theta}_A$. However, this does not hold when the true parameters are on the boundary of $\mathbf{\Theta}_A$ (Casella and Berger [2001], pg. 516). If the true parameters are on the boundary, as in the restricted tests (5) and (6), there will be a point mass at zero in the distribution of λ_γ. In the simplest case where the dimension of $\mathbf{\Theta}_A$ is one and the true parameter is on the boundary, λ_γ will be distributed as a mixture of 0.5 mass at zero and 0.5 times a χ^2_1 distribution.

Star p-values. The simpler of the two shapes, the star, is covered first. The number of stars in a graph is just the number of nodes, and therefore, for each node v, we can afford to model the distribution of the GLRT $\lambda_v = \sum_{e \in outedges(v)} \lambda_e$ for the star around v (see (2)).

Let Λ_v have the distribution of the λ_v. Λ_v is modeled as $\Lambda_v = B_v X_v$ where $B_v \sim \text{Bernoulli}(p_v)$ and $X_v \sim \text{Gamma}(\tau_v, \eta_v)$. Since all λ_e in the sum could be zero, Λ_v must have a point mass at zero, which is captured by B_v. To model the positive part of the distribution for Λ_v, the Gamma distribution is attractive since the asymptotic distribution of λ_v is the sum of independent zero inflated χ^2 distributed random variables, and the sum of χ^2 random variables is a special case of the Gamma. The log-likelihood of N independent, identically distributed samples is given by

$$l(p, \tau, \eta) = \sum_{i=1}^{N} I(\lambda_i = 0) \log(1-p) + I(\lambda_i > 0) \left[(\tau - 1) \log \lambda_i - \lambda_i / \eta - \log \Gamma(\tau) - \tau \log \eta \right]. \tag{7}$$

To estimate τ_v and η_v, direct numerical optimization of (7) is used, spanning 10 baseline days of non-overlapping 30 minute windows, for each star centered at node v. Denote the MLEs as $(\hat{p}_v, \hat{\tau}_v, \hat{\eta}_v)$. Then for an observed λ_v, the upper p-value is calculated by $P(\Lambda_v > \lambda_v) = \hat{p}_v(1 - F_\Gamma(\lambda_v | \hat{\tau}_v, \hat{\eta}_v)$ where F_Γ is the Gamma CDF.

Path p-values. Unlike stars, the large number of paths makes modeling λ_γ for each path prohibitively expensive, both in computation time and memory requirements. Instead, a model for

each individual edge is built, and p-values are then combined during the path likelihood calculation.

For each edge e, let Λ_e have the null distribution of e's GLRT scores, λ_e. Again, a zero-inflated Gamma distribution is used to model this. Now, however, it will be on a per-edge basis. Once again, this model is motivated by the fact that asymptotically, the null distribution of λ_e is a zero inflated χ^2.

Let $\Lambda_e = B_e X_e$ where $B_e \sim \text{Bernoulli}(p_e)$, and $X_e \sim \text{Gamma}(\tau_e, \eta)$, with edge specific shape τ_e and shared scale η. That is, two free parameters exist for each edge, p_e and τ_e, and a common scale parameter is assumed for all edges, η. The importance of the common scale parameter will become clear shortly. MLEs $\hat{p}_e, \hat{\tau}_e$, and $\hat{\eta}$ are estimated from non-overlapping 30 minute windows. The likelihood is similar to (7), but since each edge has its own τ_e, and a shared η, an iterative scheme has been developed which alternates between estimating η for all edges, and then, for that fixed η, estimating individual τ_e. Since each step of the iteration increases likelihood, the overall procedure increases likelihood.

Once the edge models are estimated, all of the information needed to calculate path p-values is available. Let $\Lambda_p = \sum_{e \in path} B_e X_e$. The 3-path exceedance p-value is the mixture exceedance given by

$$\begin{aligned} P(\Lambda_p > \lambda_p) &= \sum_{b_1=0}^{1}\sum_{b_2=0}^{1}\sum_{b_3=0}^{1} P(B_1 = b_1)P(B_2 = b_2)P(B_3 = b_3)P(\Lambda_p > \lambda_p | b_1, b_2, b_3) \\ &= \sum_{b_1=0}^{1}\sum_{b_2=0}^{1}\sum_{b_3=0}^{1} \left(\prod_{i=1}^{3}(1-\hat{p}_i)^{1-b_i}\hat{p}_i^{b_i}\right)\left(1 - F_\Gamma\left(\lambda_p | \sum_{j=1}^{3} b_i\hat{\tau}_i, \hat{\eta}\right)\right) \end{aligned} \tag{8}$$

where we used the fact that the sum of Gamma random variables with common scale parameters is again Gamma.

3.7 Threshold Determination

In the simulation study in Section 4, to obtain thresholds, 10 days of per-minute counts were simulated for each edge with no anomalies introduced. In real settings, such as that described in Section 5, false discovery rates cannot be measured, since any data we use may have anomalies in it. Therefore, we term them "discovery rates" instead of false discovery rates.

In either case, a 30 minute window is slid over time, offset by 10 minutes, over the 10 days of either simulated or real data, calculating the minimum p-value in each window. To achieve a (false in the case of simulation) discovery rate of 1 alarm per day, one might take the tenth smallest p-value in the resulting list of p-values. But since the windows overlap, we choose to be less conservative, by counting minimum p-values resulting from consecutive windows on the same path as a single

p-value, and finding the tenth smallest minimum p-value associated with non-consecutive windows. In this way, alarms over several overlapping windows only contribute one alarm to the threshold determination, which is exactly the way an analyst would view a series of consecutive alarms.

4 Simulation Study

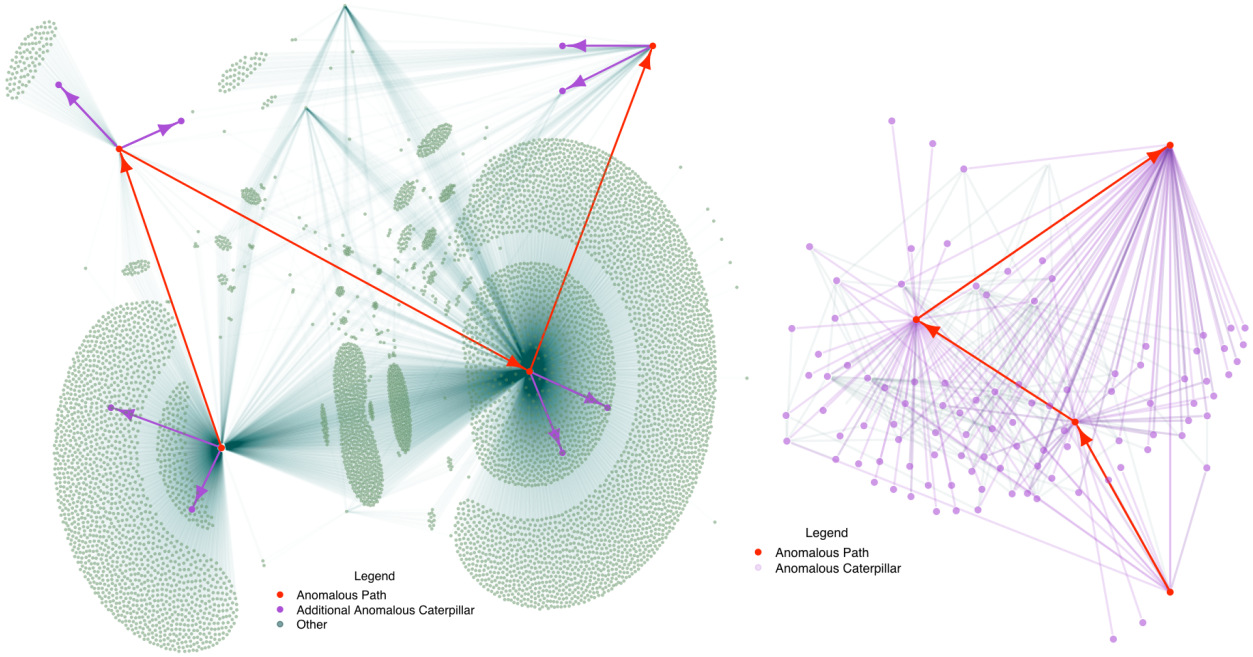
In this section a series of simulations is described, applying the graph scanning approach to a variety of models and test types. For a given anomaly type and model, the simulation involves the following general steps:

1. For each edge, estimate historic parameters from the full 30 days of LANL data (see Section 3.4)
2. Using the models estimated above, generate 30 days of simulated data, and fit models for the distribution of the λ_γ scores resulting from scanning this simulated data (see Section 3.6)
3. Obtain a p-value threshold from an additional 10 days of simulated, non-anomalous data (see Section 3.7)
4. Simulate 100 days of minute data on each edge according to the historic estimates, except for the set of anomalous edges, where the model parameters are adjusted to introduce an anomaly (more detail is provided in Section 4.1)
5. For each of the 100 days of scanning
   (a) Slide a window of length 30 minutes over the day, offsetting each consecutive window by 10 minutes
   (b) Within each window, select the edges of the entire data set for which there was at least one non-zero count in the window. This creates a subgraph of the overall graph
   (c) For this subgraph, enumerate all 3-paths, and calculate their p-values
   (d) If any path in this window has a p-value below the threshold, record all such paths, and examine no further windows for this day

The idea behind step 5(d) above is that once an anomaly is detected, the system would pass the results to an analyst. This analyst would possibly shut down the machines involved, and determine what, if any, true malicious activity was present, before allowing the machines back on the network. Therefore, these first detection graphs are the only graphs analyzed in the results section, since for any further windows in the day, the anomaly would not be present in the data after forensic analysis was performed.

4.1 Anomalous Paths

In order to better understand the system performance on anomalous paths that traverse different parts of the network, two different paths were chosen for which anomalous counts would be gen-



(a) Path/Caterpillar A (b) Path/Caterpillar B

Figure 7: Path and Caterpillar Anomalies. For each core path, the anomaly is plotted, along with the directly connected edges for context. Red edges and nodes give the core path, and additional caterpillar edges are plotted in purple.

erated. The first path (path A) is meant to be representative of paths traversing through heavily connected nodes. The second path (path B) traverses more lightly used machines. Paths A and B are depicted in Figure 7. The red path in each figure indicates the true anomalous edges. The purple edges are part of a separate caterpillar anomaly, discussed in Section 4.2. All other green edges in the plot are either from or to a node involved in the anomalous path and are provided for context.

	Node 1	Node 2	Node 3	Node 4	CGS
Path A	2640 / 240	141 / 89	5131 / 1446	95 / 65	220640
Path B	6 / 13	13 / 23	6 / 43	22 / 55	8931

Table 1: In / Out Degrees of Nodes in Anomalous Paths

The number of in and out edges for each node (degree) in each of the two path types is summarized in Table 1. In addition, the Containing Graph Size (CGS) is given, which is the number of edges in the union of all 3-paths in the graph which contain at least one of the truly anomalous edges. Thus, the CGS is the number of unique edges that are part of any 3-path that passes through at least one of the truly anomalous edges. Observe that the CGS of Path A is roughly 39.4% of the entire edge set, while path B's CGS is only 1.6%.

Anomalous Parameter Settings To simulate a path anomaly, the estimated parameters in each

edge of the anomalous path are modified before simulating, but the historic parameters for all other edges are used.

In the OMM simulation, the anomaly was created by letting $p_{01}^{anom} = \hat{p}_{01} + 0.2$ on each of the anomalous path edges. This increase was arrived at after consulting with cyber-security experts, whose intuition was that a likely hacker behavior could be to transition to the active state once every two minutes. A more conservative approach was taken, by inserting a one in five minute anomaly. In the HMM simulations, three types of anomalies are summarized in Table 2. The high state mean was raised in the M and B anomaly types, reflecting the fact that a hacker may act in a way that increases the historic mean by one count per minute. All parameters not mentioned in each type are left at their historic MLEs. To be specific, the historic and anomalous parameters for each path type are given in Table 3.

Type	Anomalous Parameter Change
P	$p_{01}^{anom} = \hat{p}_{01} + 0.2$
M	$\mu^{anom} = \hat{\mu} + 1$
B	$p_{01}^{anom} = \hat{p}_{01} + 0.2, \mu^{anom} = \hat{\mu} + 1$

Table 2: HMM Anomalous Parameter Settings

Model	Path	Edge	$\hat{p}_{01}$	p_{01}^{anom}	$\hat{p}_{10}$	$\hat{\mu}$	μ^{anom}	$\hat{s}$
OMM	A	1	0.03	0.23	0.50	NA	NA	NA
		2	0.01	0.21	1.00	NA	NA	NA
		3	0.02	0.22	0.97	NA	NA	NA
	B	1	0.07	0.27	0.97	NA	NA	NA
		2	0.02	0.22	0.90	NA	NA	NA
		3	0.02	0.22	0.99	NA	NA	NA
HMM	A	1	0.01	0.21	1.00	3.44	4.44	432.84
		2	$3e$-4	0.20	$2e$-3	0.47	1.47	123.30
		3	0.02	0.22	0.97	3.43	4.43	19.07
	B	1	0.05	0.25	0.96	0.32	1.32	121.82
		2	0.04	0.24	0.67	0.50	1.50	150.84
		3	0.22	0.42	0.93	0.35	1.35	121.82

Table 3: Changes to Model Parameters for Anomalous Paths. $\hat{}$ indicates historic parameter estimates, *anom* indicates anomalous parameter settings.

Path Scanning Results The results of using paths to detect the simulated path anomalies described above are briefly summarized. For a more detailed analysis of the simulation results, refer to the Supplementary Material.

For each model, a box plot of the time to first detection is given in Figures 8 and 9. Recall that for each of 100 days, the scanning was performed on 30 minute windows, slid by 10 minutes, until any path's p-value was smaller than the false discovery rate threshold (as determined by the method described in Section 3.7). This first daily detection time is the statistic plotted in the box plots. The value to the right of each box plot, denoted PD, gives the percentage of the 100 days in which any window was detected.

In Figure 8, the time to first detection box plot for the OMM simulations is given. In both Path A and Path B, at least one path was detected in each of the 100 days. Detection occurred more quickly in the day for Path A than for Path B, and reflects the fact that there were many more paths traversing Path A than Path B, and therefore there are more chances per time window to detect at least one edge in Path A.

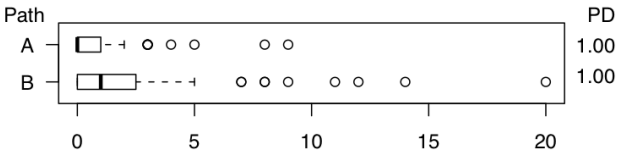


Figure 8: OMM Time to First Detection. The x-axis represents the time window of first detection, in multiples of ten minutes.

Similar box plots for the HMM simulation time to first detection are given in Figure 9. The columns on the left of each plot indicate the combination of type of anomaly introduced (Anom) (see above), parameter combination being tested (Test) (see Section 3.5), and which path was truly anomalous (Path) (see above).

Figure 9 illustrates that Path A anomalies were detected rapidly on every example except for the M anomaly. To see why this is so, refer to Table 3, and recall that in the M anomaly, p_{01} was not anomalous. The historic p_{01} is very low on Path A. In addition, the historic μ is relatively high on this path. These two parameters work together to make the detection of an anomalous mean difficult.

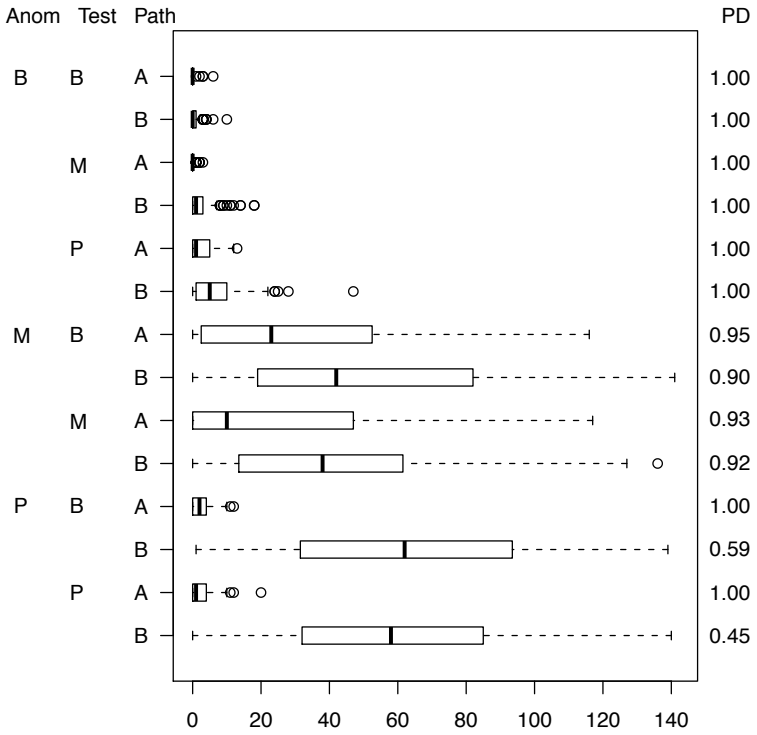


Figure 9: HMM Time to First Detection. The x-axis represents the time window number of first detection, in multiples of ten minutes.

Nevertheless, Path A was consistently detected earlier than Path B on the same anomaly/test combination. Again, this is explained by the fact that more paths intersect Path A, and therefore there are more chances to test paths containing some of the edges in Path A than Path B. Path B took far longer to detect, on average, than Path A on the P anomalies. This is in part due to the historically high p_{01} on Path B, as compared to Path A (again, see Table 3). In addition, edge 2 on Path B had a relatively low p_{10}. Therefore, if the edge enters the high state, it stays there for

longer periods of time, and there is less chance to observe low-to-high transitions in the window. For a more detailed examination of the interplay between anomaly type, test type, and path, see the Supplementary Material.

4.2 Other Anomalous Shapes: Stars and Caterpillars

In Section 2.3, using star windows to scan is discussed. Here, a star anomaly is described. Choosing one of the moderate out-degree nodes from Path B, namely Node 3 in Table 1, a B Type anomaly is introduced on every out-edge from that node. Both path and star scanning is performed, and the results are briefly described below and in more detail in the Supplementary Material.

In addition to a star anomaly, two types of caterpillar anomalies are introduced. Recall from Figure 2 that the caterpillar shape has a path at its core, and additional edges emanating from core nodes. The first caterpillar, denoted caterpillar A, has Path A as its core path, along with two additional anomalous edges emanating from each path node. Referring back to caterpillar A in Figure 7, the red edges form the core path of the caterpillar, but now the purple edges are included to form the caterpillar anomaly. This shape reflects a hacker behaving in a subtle way, by changing only a small number of edges around a path that is embedded in a much larger subgraph.

The second caterpillar, denoted caterpillar B, is also shown in Figure 7, and consists of Path B at its core, along with every out edge around this path's nodes, given in purple. This represents a hacker making many more anomalous edges per hop in the traversal, but the path is embedded in a much smaller subgraph than caterpillar A. These anomalies were designed to compare the performance of paths and stars on a mixed path-star anomaly. There are a total of 43, 11, and 174 edges in each of the anomalies, Star, Cat A, and Cat B, respectively.

In simulation, only the HMM model was used on the star and caterpillar anomalies. The anomaly introduced increased both p_{01} and μ, and the test was for an increase in both of these parameters. However, for both caterpillar and star anomalies, both star and 3-path shapes were used to scan.

Results of Path Scanning on Star and Caterpillar Anomalies In Section 4.2 simulations involving star and caterpillar anomalies were discussed. Visualizations of the scan results for each type of anomaly are given here, and a more complete analysis is presented in the Supplementary Material.

In Figure 10, a visualization produced by using a star-shaped window to scan the Caterpillar A anomaly is given. One can see that when using a star shaped scan most of the anomalous edges were missed, and many false edges were detected.

A detection in this setting corresponds to the union of every scan window which had a p-value smaller than the FDR threshold. These shapes may overlap on a set of edges. While stars do not overlap, paths do, and so for each detected graph the number of times each edge appears in any detected path can be collected, and this count can then be used to color edges in a heat map of the detection.

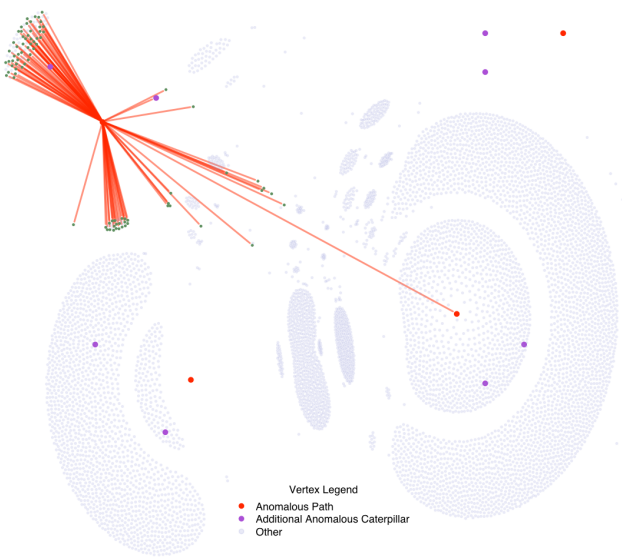
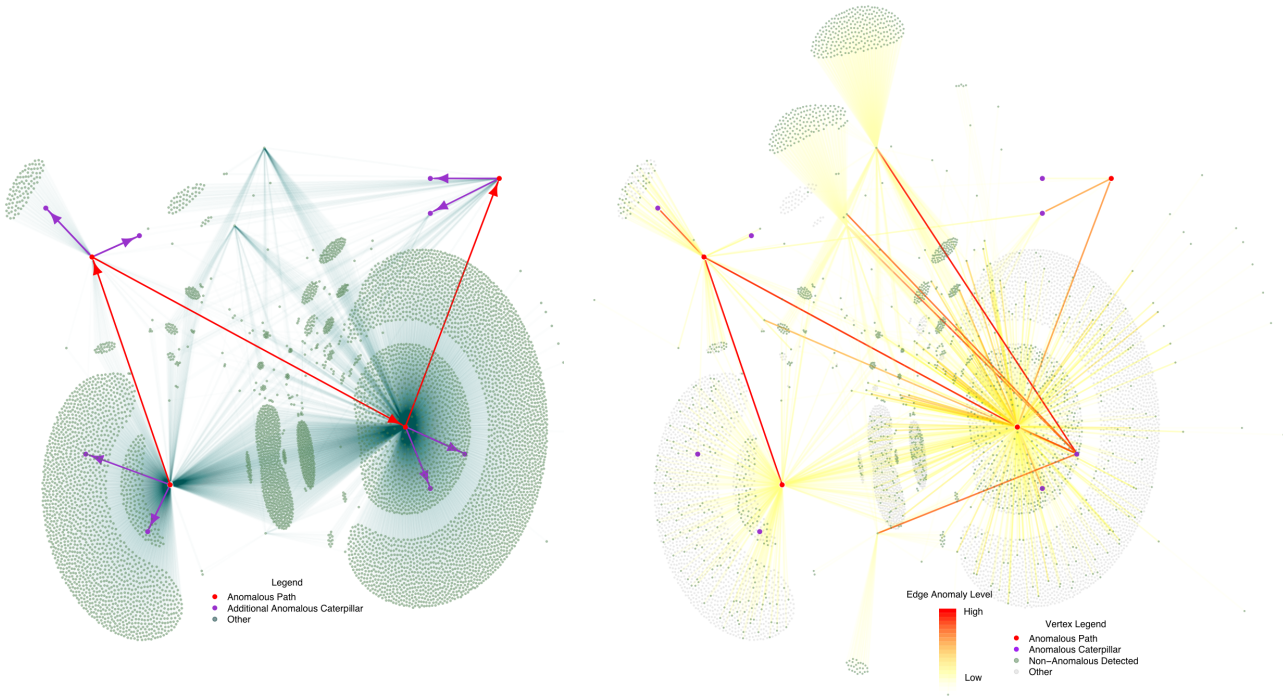


Figure 10: Star Scanning Results for Caterpillar A. Detected edges are plotted in red.

A heat map resulting from using a path shape on the anomaly given by Caterpillar A is presented in Figure 11. On the left, caterpillar A is plotted, embedded in its 1-hop containing graph (i.e., all edges emanating from the nodes in the caterpillar). On the right, the path scan



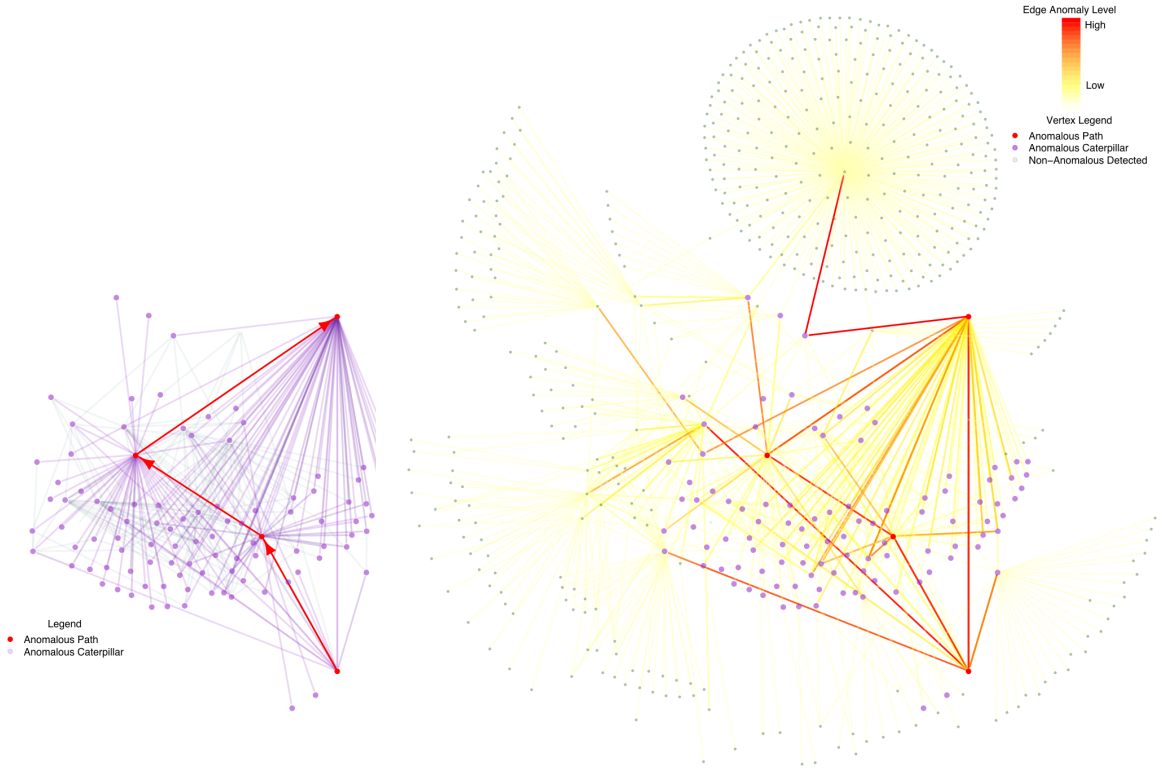
(a) Caterpillar A (b) Detection Heat Map

Figure 11: Anomaly Graph and Heat Map For Caterpillar A. The true anomaly is given on the left, with anomalous edges colored red and purple. The detected heat map is displayed on the right, with darker red indicating more evidence of an anomaly.

heat map of a single detected window is given. In contrast to the star results in Figure 10, the core path is now brightly colored, as these edges were detected very frequently. In addition, while some

edges may be dim (for this detection), every true anomalous edge is present in this detection graph. These colors not only give the analyst an ordering of importance of the edges, but also provide an overall view of the structure of the anomaly. It additionally highlights the ability of paths to form more general shapes of detection than just the core shape.

In Figure 12, a similar heat map of the path scanning results on the caterpillar B anomaly is provided. The core path was well detected, and most of the total anomaly was detected. There is a large star in the upper right of false detections. This is due to a highly connected node sharing an edge with the true anomaly, allowing paths through that node to intersect the true anomaly.



(a) Caterpillar B

(b) Detection Heat Map

Figure 12: Anomaly Graph and Heat Map for Caterpillar B. The true anomaly is given on the left, with anomalous edges colored red and purple. The detected heat map is displayed on the right, with darker red indicating more evidence of an anomaly.

The two detections in Figures 11 and 12 are fairly remarkable. Of the greater than 90,000 potentially anomalous edges under examination, the method has singled out around two dozen edges which, in a real network security setting, would be candidates for further examination. These types of massive reductions are required in order to efficiently capitalize on the limited time that network security practitioners can devote to any single detection, and gives us confidence that this method presents a viable option for operational needs.

5 Real Network Detection

Since the goal with this work is a system that runs in real time, on real networks such as LANL's internal network, we considered it an important milestone to run a path scan on real data from such a network. Therefore, in this section a path scan analysis of data contained in LANL's historic data archives is described.

5.1 Implementation Details

The HMM model was used, and the parameters were estimated from real data, starting January 30, 2010, and ending 30 days later, as described in Section 3.4. A test for an elevation in p_{01} was performed. Initially, a test of both parameters was attempted, but we encountered several numerical problems with testing the high-state mean that could only be resolved with a more custom model for this data. In addition, testing for only a p_{01} change had good performance in simulation, especially when the mean was also anomalously high (see the Supplementary Material for a discussion of this fact).

Since simulated data was used to set p-value thresholds in the simulations, new thresholds were required when preparing to scan on real data. Therefore, the next 10 days of data, starting March 2, and ending March 12, were used to obtain these thresholds, using a discovery rate of one detection per day. That is, T was set such that during these 10 days there were 10 distinct "events" that had a path with p-value less than T, as described in Section 3.7. Finally, the next 20 days were scanned using 3-paths.

Completely unestimated (new) edges did arise in this data set. For this example, these new edges were used in enumeration, allowing estimated edges to be "bridged" by these new edges in the paths. But the data on these new edges did not contribute to the path GLRT score. A modeling approach that allows for an explicit treatment of new edges is a topic of ongoing work.

In these 20 days, 38 unique detections occurred, which is not unreasonable for this example. One would expect 20 detections, but the larger number of detections can be attributed to estimation error in setting the threshold, random fluctuation in the number of detections, and/or some deterioration of the model fits over time. In practice, a larger sample would be used for setting the threshold, and these models would be updated over time.

5.2 Description of One Detection

While many of these detections look interesting, we choose to describe the most anomalous one, i.e. the detection that achieved the minimum p-value in the 38 days, in detail. A heat map of this

detection, which occurred on March 22, 2010, is provided in Figure 13.

In this figure, a star of 11 nodes around a central node is given, along with a 2-path (red) beginning at the central node. This central node is a calendaring server, and the star nodes around it are user machines making connections to it to get the updated meeting schedule. The red edge leading out from the calendaring server is an edge to a user machine (the purple node). The edge leading out from this user machine is an email server.

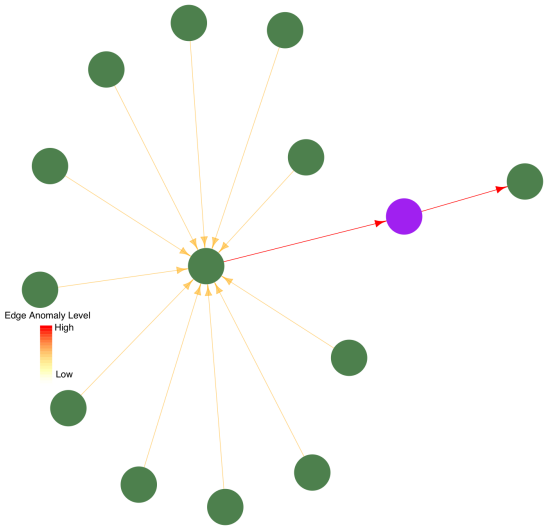


Figure 13: Real Detection Heat Map

On March 22, at around 11:00 am, this graph was detected as anomalous. Each of the edges leading to the calendaring server were identified once in the detected graph, and the two red edges were detected 11 times. This implies that the 11 3-paths starting at each star node all passed through both red edges.

When a forensic analysis of this graph was conducted, two relevant facts emerged. First, the rate of counts on the two red edges increased significantly, while the edges leading into the star did not. This indicated an embedded anomalous 2-path in the 3-paths, which is apparent in Figure 13. Second, it was determined that the purple node changed significantly. Specifically, the purple machine's user changed.

Since the user changed, the settings of applications which accessed the network from this computer changed. While this event could be explained by normal network usage, it is nonetheless a very promising detection. Without the legitimate user change, this would be an extremely interesting anomaly, possibly indicating the presence of a hacker, and one that our security team would investigate thoroughly. Since the goal was to implement a practical monitoring system that detects just such anomalies, this finding is very encouraging.

6 Conclusions & Future Work

A method for detecting anomalous activity is described, where data is defined over time on edges in an underlying graph structure. The need for anomaly detection in this setting is motivated with the example of a hacker traversing a computer network. Attacks can be very localized, and so a method of windowing locally in the time $\times$ graph space is introduced. In each window, a

scan statistic indicating whether or not the data in this local window is behaving according to a historic model is calculated. Many attacks have, at their core, a path that is the result of a hacker traversing through the network. Therefore, we have introduced k-paths as a versatile type of local graph window for detecting relevant anomalies.

An online system for analyzing streaming data in real time is described. The system is prototyped, and has been run on a variety of simulated and real data sets. The underlying algorithm for enumerating paths is extremely efficient, i.e., capable of enumerating many billions of local windows in the graph in seconds. In addition, the results of simulations and a real data example are given. The simulations provide insight into system performance on a variety of different anomalies and testing schemes. The real-data example is exciting, since we have detected the very activity we set out to detect. Heatmaps that should aid in the forensic investigation of detected graphs are presented. This system will be instrumented on LANL's computer networks, and we are confident that it will increase the security of our communications.

Most of our work so far has been focused on the general framework of scanning. Further work is focused on developing models that better represent computer edge data. The OMM and HMM presented here are not entirely sufficient to model the network data to our satisfaction. One of the most important features of the further modeling work involves treatment of new and sparse edges through the use of a hierarchical model.

In addition, the data exhibit daily and weekly patterns. As seen in Figure 1, the middle part of the work day has higher counts than at night. Different schedules on different days can also be seen.

Thus, the parameters of the model should also be allowed to change smoothly through the day, similar in spirit to the diurnal and weekly patterns modeled on telephone call networks presented in Lambert and Liu [2006]. While there are many areas for improvement, we are confident that the basic approach of path scanning will be very effective in identifying anomalous activity within computer networks.

References

J. Baker. The dragon system–an overview. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 23(1):24–29, 1975.

L.E. Baum and JA Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc*, 73(3):360–363, 1967.

L.E. Baum and GR Sell. Growth functions for transformations on manifolds. *Pac. J. Math*, 27(2): 211–227, 1968.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.

N. Brownlee, C. Mills, and G. Ruth. Traffic flow measurement: architecture (rfc 2722), 1999.

G. Casella and R.L. Berger. Statistical inference. *Duxbury Press*, 2001.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

M. Collins and M. Reiter. Hit-list worm detection and bot identification in large networks using protocol graphs. In *Recent Advances in Intrusion Detection*, pages 276–295. Springer, 2007.

H Djidjev, G Sandine, CB Storlie, and S Vander Wiel. Graph based statistical analysis of network traffic. In *Proceedings of the 9th Workshop on Mining and Learning with Graphs (MLG 2011)*, 2011.

A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Verlag, 2001.

S. Forrest, S.A. Hofmeyr, A. Somayaji, T.A. Longstaff, et al. A sense of self for unix processes. In *IEEE Symposium on Security and Privacy*, pages 120–128. IEEE COMPUTER SOCIETY, 1996.

Joseph Glaz, Joseph Naus, and Sylvan Wallenstein. *Scan Statistics*. Springer, 2001.

N.A. Heard, D.J. Weston, K. Platanioti, and D.J. Hand. Bayesian anomaly detection methods for social networks. *Annals*, 4(2):645–662, 2010.

E.D. Kolaczyk. *Statistical Analysis of Network Data: Methods and Models*. Springer, 2009.

Martin Kulldorff. A spatial scan statistic. *Communications in Statistics- Theory and Methods*, 26 (6):1481–1496, 1997.

D. Lambert and C. Liu. Adaptive Thresholds: Monitoring Streams of Network Counts Online. *Journal of the American Statistical Association*, 101(473):78–88, 2006.

D. Lambert, J.C. Pinheiro, and D.X. Sun. Estimating Millions of Dynamic Timing Patterns in Real Time. *Journal of the American Statistical Association*, 96(453), 2001.

C.R. Loader. Large-deviation approximations to the distribution of scan statistics. *Advances in Applied Probability*, 23(4):751–771, 1991.

Q. Lu, F. Chen, and K. Hancock. On path anomaly detection in a large transportation network. *Computers, Environment and Urban Systems*, 2009.

G. Lyons. Network Working Group P. Amsden Request for Comments: 2124 J. Amweg Category: Informational P. Calato S. Bensley, 1997.

B. Mukherjee, L.T. Heberlein, K.N. Levitt, et al. Network intrusion detection. *IEEE network*, 8 (3):26–41, 1994.

J.I. Naus. Approximations for distributions of scan statistics. *Journal of the American Statistical Association*, 77(377):177–183, 1982. ISSN 0162-1459.

C.C. Noble and D.J. Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2003. ISBN 1581137370.

P. Phaal, S. Panchen, and N. McKee. Inmon corporations sflow: a method for monitoring traffic in switched and routed networks (rfc 3176). Technical report, Technical report, Internet Engineering Task Force (IETF), 2001.

W. Pohlmeier and V. Ulrich. An econometric model of the two-part decisionmaking process in the demand for health care. *The Journal of Human Resources*, 30(2):339–361, 1995. ISSN 0022-166X.

Carey E. Priebe, John M. Conroy, and David J. Marchette. Scan statistics on enron graphs. In *Workshop on Link Analysis, Counterterrorism and Security at the SIAM International Conference on Data Mining*, Newport Beach, CA, 2005.

LR Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

D.Y. Yeung and Y. Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36(1):229–243, 2003.